Carnegie Mellon
Software Engineering Institute

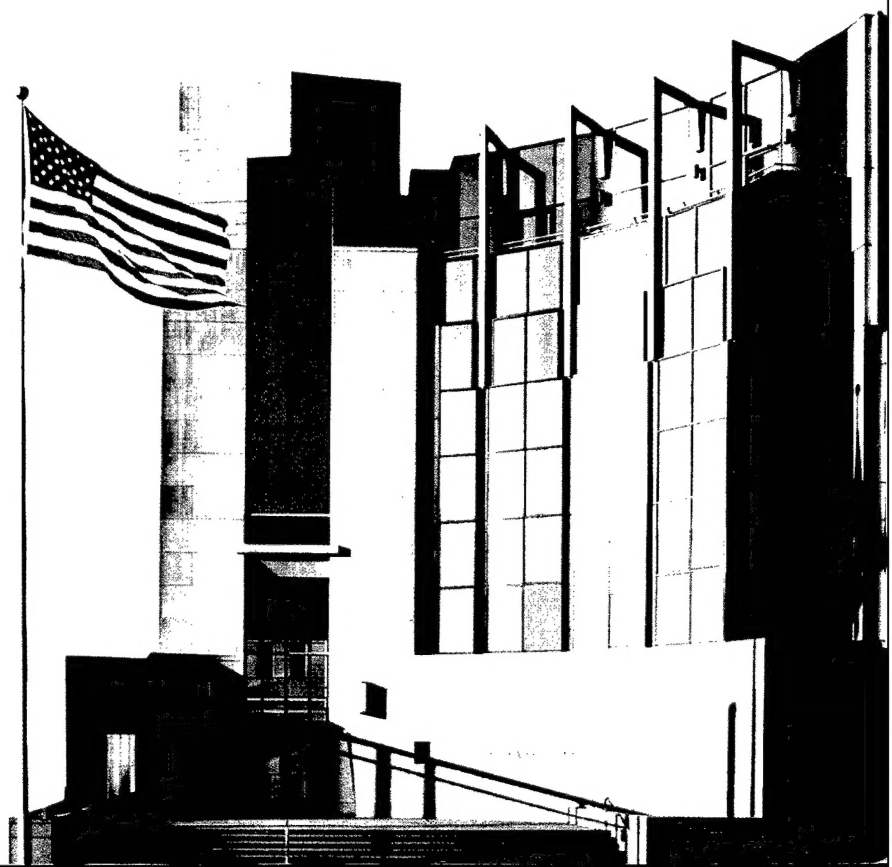
Experiences in Implementing Measurement Programs

Wolfhart Goethert
Will Hayes

November 2001

Software Engineering Measurement and Analysis Initiative

Technical Note
CMU/SEI-2001-TN-026



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Experiences in Implementing Measurement Programs

Wolfhart Goethert
Will Hayes

November 2001

Software Engineering Measurement and Analysis Initiative

Technical Note
CMU/SEI-2001-TN-026

printed 11/27/2001 3:38 PM version number 2/ LHEINZ

Contents

Abstract		**vii**
1	**Background**	**1**
2	**Overview of the Measurement Programs**	**3**
	2.1 Case 1: Measurements Across a Global Enterprise	3
	2.2 Case 2: Assessing the Impact of Software Process Improvement	3
	2.3 Case 3: Enterprise Performance Management, a Local Perspective	4
3	**Description of the Methodology**	**5**
4	**Identification of Business Goals**	**8**
5	**Indicators**	**9**
	5.1 Identification of Indicators	9
	5.2 Classification of Indicators	10
	5.3 Number of Indicators	11
	5.4 Using the Full Set of Indicators	11
6	**Definitions**	**13**
	6.1 Indicator Definition Template	13
	6.2 Definition Checklist	15
	6.3 Precise Definitions	17
7	**Data Elements or Measures**	**19**
	7.1 Use of the Data Elements: Etiquette	19
	7.2 Collection of Data	19
	7.3 The 100% Solution May Not Be Feasible	20

8 Lessons Learned **21**
8.1 Pilot Implementations 21
8.2 Implementation Time Frame 22
8.3 Automation/Tools 22
8.4 Using Analysis Indicators 22
8.5 Motivating the Wrong Behavior 23
8.6 Summary of Lessons Learned 24
8.7 Conclusion 26

References **27**

Appendix A: Indicators For Case 1 **29**

Appendix B: Indicator Definition Template **32**

List of Figures

Figure 1: Measurement Program Starts and Successes 1
Figure 2: Goal-Driven Software Measurement 6
Figure 3: Goal-Driven Software Measurement Workshop 7
Figure 4: Types of Indicators 10
Figure 5: Indicator Template 14
Figure 6: Definition Checklist 16
Figure 7: Adapted Staff-Hour Checklist 16
Figure 8: Developing a Checklist for a Project's Start and End Dates 17
Figure 9: Date Definition Checklists 18
Figure 10: Example of a Start and End Date Checklist 21
Figure 11: Changes per Release Number 23
Figure 12: Enterprise Profile Example 31
Figure 13: Indicator Template 32
Figure 14: Modified Indicator Template 35

List of Tables

Table 1:	Focus of Cycle Time Indicator	9
Table 2:	Metrics Collection Form [Augustine 99]	15
Table 3:	Data Access Recommendations	19

Abstract

Despite significant improvements in implementing measurement programs for software development in industry, data collected by Rubin Systems show that a large percentage of metrics programs fail. This technical note describes some useful lessons learned at a number of organizations that have implemented measurement programs using the Goal-Driven Software Measurement methodology. It includes a description of the methodology, a discussion of the challenges, obstacles, and their solutions, an initial set of indicators and measures, as well as some artifacts (such as templates and checklists) that we have found to enable successful implementations. The main motivation of this technical note is to provide some practical advice and guidelines for planning and implementing measurement programs.

1 Background

Data collected by Howard Rubins of Rubin Systems, Inc. show that four in five metrics programs fail to succeed [Pitts 97]. Here, *success* is defined as a measurement program that lasts for more than two years and that impacts the business decisions made by the organization.

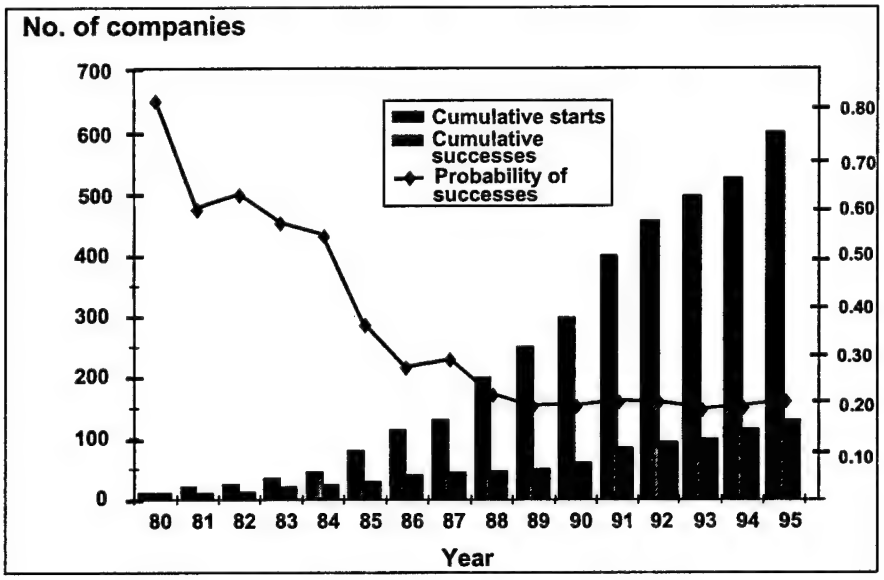


Figure 1: Measurement Program Starts and Successes

The primary reasons that metric programs fail are not due to technical issues but rather due to organizational issues [Rubins 92]:

- not tied to business goals
- irrelevant or not understood by key players
- perceived to be unfair, resisted
- motivated wrong behavior
- expensive, cumbersome
- no action based on the numbers
- no sustained management sponsorship

A successful measurement program is more than collecting data. The benefit and value of doing software measurement comes from the decisions and actions taken in response to analysis of the data, not from the collection of the data [Zubrow 98]. One of the challenges faced by measurement professionals in large complex organizations (such as those

developing and maintaining major software systems) is the fact that so many opportunities for measurement exist. The search for the "right" measures can easily become overwhelming when the selection is not driven by the information requirements to be addressed by the measures. For measurement to be cost effective, it must be designed and targeted to support the business goals of the organization. In their survey of organizations with "reputations for having excellent measurement practices," Rifkin and Cox observed that this tie between measures and goals is one of the characteristics that differentiate successful organizations from the rest [Rifkin 91]. The goal-driven software measurement process produces measures that provide insights into important management issues as identified by the business goals. Since these measures are traceable back to the business goals, the data collection activities are better able to stay focused on their intended objectives. Hence measurement and analysis is planned to support goals; no measurement is done just to collect data for the sake of collection alone.

In this paper, we summarize a number of different case studies, which illustrate the application of goal-driven measurement in diverse settings. These organizations had very different goals for their measurement programs, and chose to focus on measures that were uniquely suited to their needs. The Software Engineering Institute (SEI) provided assistance to these organizations in implementing their measurement programs. Using artifacts and lessons learned from these organizations; we will discuss the issues and challenges faced in implementing these measurement programs. Despite the rather obvious differences in the needs of these organizations, the impediments they faced are very similar. In the following sections, we summarize the various steps in the measurement process and provide advice for improving the success of a measurement program.

2 Overview of the Measurement Programs

2.1 Case 1: Measurements Across a Global Enterprise

The purpose of this measurement program was to establish a system of uniform measures across a global enterprise. It included designing measures that supported global business concerns and creating an organizational infrastructure that spanned diverse geographic locations and cultural milieus. The business units involved in this work had different business concerns, processes, native languages, and cultures. These differences were so pervasive that it was sometimes difficult to establish common definitions for even very basic terms, such as the word "project" for example. What was identified as a *project* in one business unit, might be called a *task* in another unit. Some business units had multiple tasks that made up projects, while others preferred to call each of those tasks a project. The global scope exacerbated many already difficult technical problems, such as generating definitions that could be applied consistently, and normalizing data for comparison purposes.

Among the advantages that this organization was trying to achieve with their measurement program were

- the ability to answer questions about the enterprise (For example, are we getting better or getting worse; is an enterprise-wide improvement program having an effect?)
- the ability to evaluate new technologies, methods, and practices by
  - collecting identical measures to enable meaningful comparisons and trend analysis
  - creating a large pool of project data from which similar projects can be chosen for comparison purposes
- the establishment of a visible ongoing enterprise focus for software engineering excellence

The participating business units involved were located around the globe: Tokyo, Singapore, Hong Kong, India, Argentina, and the United States.

2.2 Case 2: Assessing the Impact of Software Process Improvement

The purpose of this measurement program was to assess the impact of investment in Software Process Improvement (SPI). As a consequence of the ongoing implementation of SPI activities, the schedule, cost, and quality of future software projects were ***expected*** to be significantly better than previous efforts. The indicators in this measurement program were developed to understand, influence, and communicate the ***actual*** benefits of software process improvement to completed software projects. The defined measures were to serve as a means

of focusing attention on what was important rather than on the many other interesting aspects of software development.

2.3 Case 3: Enterprise Performance Management, a Local Perspective

The purpose of this measurement program was to support management in workload balance and effective project management in the context of an ongoing process improvement program. Standardization of measurement across the organization had been a major theme, but the alignment of more "local" performance objectives was needed to reconcile perceived conflicts between what the customer demands and what "corporate" requires. This organization's workload consisted of maintenance and enhancement activities across a portfolio of major systems with a diverse set of users and stakeholders. Strategic planning was conducted at the enterprise- and business-unit- levels. Relating the performance of small groups of technical staff to the mission of the enterprise was the ultimate goal for this very large organization.

3 Description of the Methodology

Through a series of workshops, these organizations used the goal-question-(indicator)-metric (GQ[I]M) methodology to define a set of measures related to their business goals. The "I" in parentheses distinguishes the GQ(I)M methodology from the closely related GQM methodology introduced and described by Basili and Rombach [Basili 88, Basili 89, Rombach 89].

The steps of the GQ(I)M approach are organized into three general sets of activities:

1. goal identification
2. indicator identification and the specification of data needed
3. infrastructure assessment and action planning to guide the implementation

In the goal-driven software measurement methodology, business goals are translated into measurement goals (Basili and Weiss, 1984; Briand et al., 1996) by first identifying high-level business goals and then refining them into concrete operational statements with a measurement focus. This refinement process involves probing and expanding each high-level goal by using it to derive quantifiable questions whose answers would assist managing the organization. The questions provide concrete examples that can lead to statements that identify what type of information is needed. In originally devising this measurement scheme, Basili emphasized the importance of having a purpose for the measurement data before selecting data to collect. Without a purpose, we cannot know what the "right" data would be.

In our elaboration of Basili's methodology, we have added an intermediate step to assist in linking the questions to the measurement data that will be collected. The importance of linking data to the questions they answer is clear in the success Basili has had with the GQM approach. Our experience suggests that identifying questions and measures without visualizing an indicator is often not sufficient to get a successful measurement program started. The displays or reports used to communicate the data (called indicators in our variation of the GQM methodology) are a key link that can determine the success or failure of a measurement program. These indicators serve as a requirement specification for the data that must be gathered, the processing and analysis that must take place, and the schedule by which these activities occur.

Following the specification of indicators, an action-planning step is carried out. First, the existing data collection and measurement activities within the organization are analyzed to avoid duplication and to identify gaps. Priorities, in terms of data to gather to produce the indicators, are assigned. Then tasks are defined to take advantage of existing activities and to address the gaps. Part of the plan also addresses the need for the measurement activities to

evolve as the organization's goals change or as new insights are gained using the measurement program.

The goal-driven software measurement approach is described in the SEI's Goal-Driven Software Measurement Guidebook [Park 96]. Figure 2 depicts the general approach we used to derive the indicators and measures.

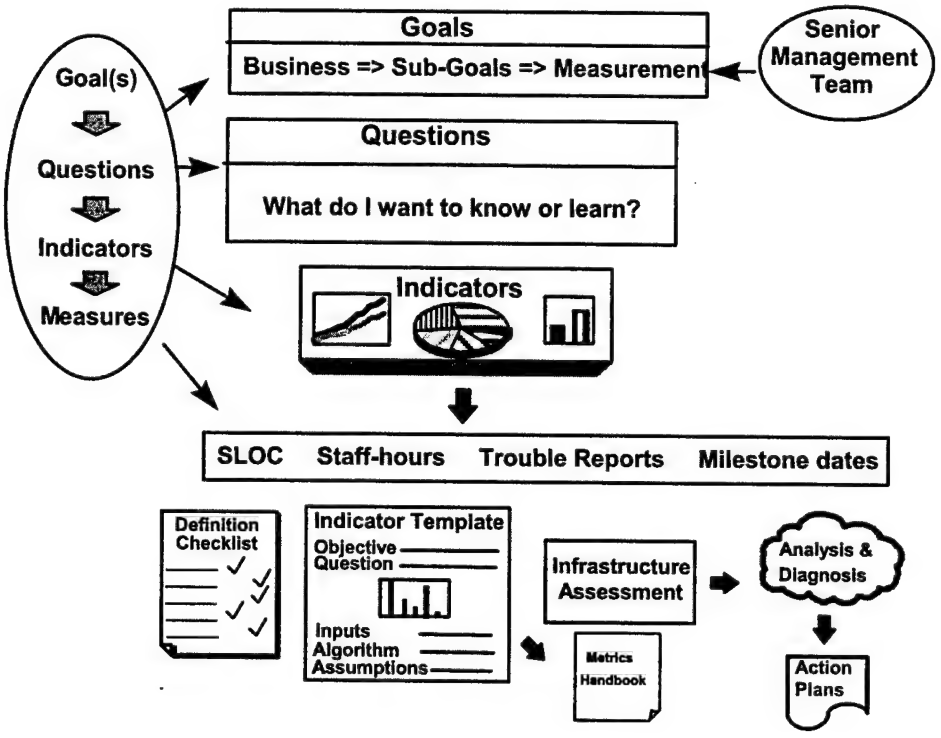


Figure 2: Goal-Driven Software Measurement

The goal-driven software measurement methodology was implemented in a 10-step course/workshop as shown in Figure 3. Tailored versions of the course are presented in workshop format in an industrial training approach.

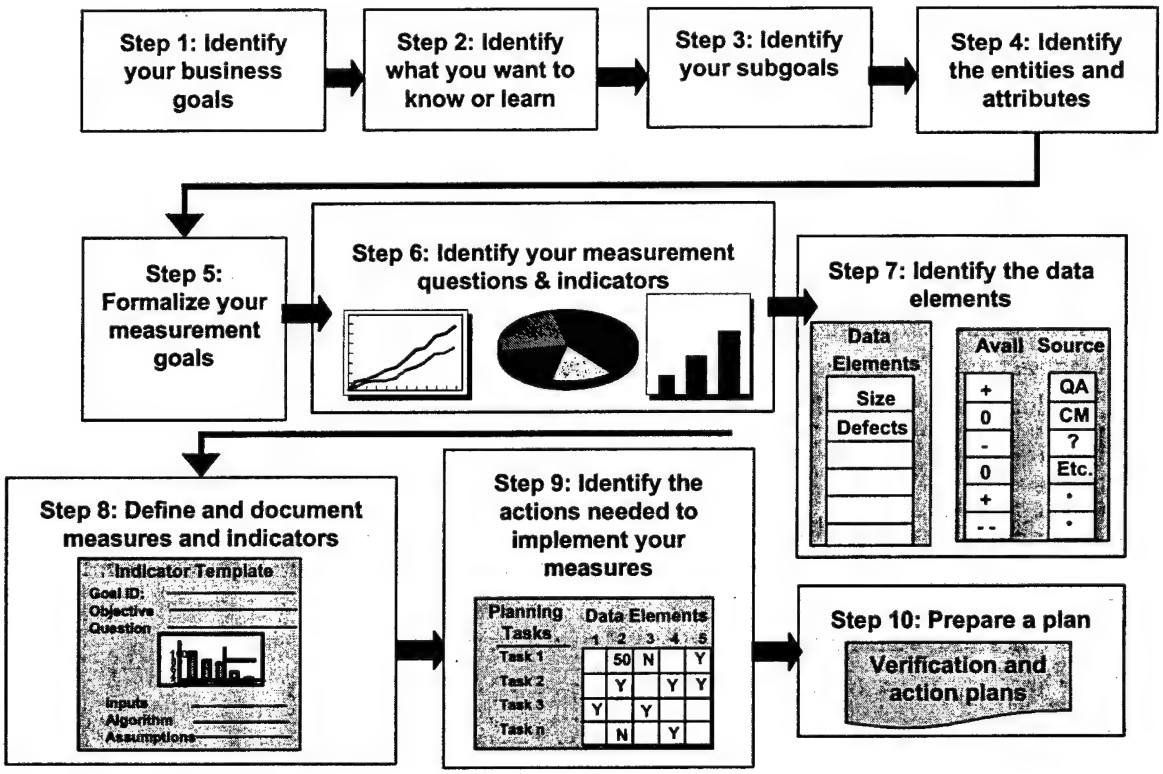


Figure 3: Goal-Driven Software Measurement Workshop

Advice: Use the GQ(I)M methodology to identify your indicators and measures to ensure traceability back to the business goals.

4 Identification of Business Goals

Clear specification of one or more business goals is a necessary input to a meaningful measurement program. These goals serve to identify the purpose for work underway in the organization. When these goals are well articulated, they beg questions that lead us to evaluate success or failure with regard to the purpose of the work rather than some arbitrary characteristic of the work itself.

In Case 1, the general business goals were articulated by the Chief Technology Officer (CTO). He wanted to measure progress towards the following corporate improvement goals:

- Increase productivity by a factor of 2 over 5 years.
- Improve quality by a factor of 10 over 7 years.
- Improve predictability to within 5% over 7 years.
- Reduce development time by 40% over 7 years.
- Reduce maintenance effort by 40% over 7 years.

In Case 2, the entire software process improvement initiative was guided by the vision of software excellence articulated by senior management that explicitly describes the attributes of the desired state that are considered essential for the foreseeable future: world-class cycle time, productivity, and quality. The measurement program was to serve as a means of focusing attention on what is important rather than the many other interesting aspects of software development that could be measured.

In Case 3, establishing a common basis for comparing information across a widely distributed organization was a major concern. From the perspective of the organizational sub-unit, their top priority was to report compelling information that accurately reflects the work being done. From the perspective of the sponsor, the expressed priorities were to support enterprise-wide performance management and for using measurement to support the transfer of process improvement suggestion across the enterprise.

Advice: Clearly specify the goal that is being addressed.

5 Indicators

5.1 Identification of Indicators

In goal-driven measurement, the primary question is not *What metric should I use*, but *What do I want to know or learn*? Starting with each organization's corporate goals, we conducted workshops with representatives of the organizations to work through the GQ(I)M methodology. This 10-step workshop is illustrated in Figure 3 (above). Our experience shows that it is much easier to postulate indicators and then identify the data items needed to construct them, than it is to go directly to the measures. Starting with the raw data (measures or data elements) and creating an indicator can lead to convenient or elegant displays that incorporate the data but fail to answer the questions that drove the data collection. With an indicator specified, the information to be derived from the raw data has been articulated, and we are better able to construct indicators that answer the questions we care about. Also, an indicator or graph is easy to "think about" and "talk to" when you are getting input from others.

Once the indicators have been identified, we found it extremely useful to review the unique focus of each indicator. Illustrated in Table 1, is the work done in Case 2 for one of their indicators.

Table 1: Focus of Cycle Time Indicator

Indicator	Question Addressed	Expected Impact of SPI
Cycle Time	What is the trend in the number of calendar days typically used by our projects to deliver a software feature (i.e., historical schedule duration)?	The average number of days to implement a feature should decrease as a result of SPI. The greatest impact should be seen in large projects.

The workshop participants were also asked to visualize success and then answer the following questions:

- What are you going to do with the information?
- What decisions are going to be driven with this data?

The selection of indicators was also driven by a number of other factors in addition to the things we would like to know about each goal. These included

- Who is the audience for the indicators?
- What should be the total number of indicators?
- Can the indictors be interpreted correctly?

- Do the indicators provide an accurate and high-level view?
- Could you collect the data in your organization? Are there major barriers?

Advice: Maintain traceability from the indicators back to the business goals. If questions arise later about intent, you will be able to look back to the origins and provide implementation decisions that are consistent with your business objectives.

5.2 Classification of Indicators

Once the organizations had defined their goals, we found many of them had difficulty deciding how to tell if or when their business goals had been achieved. While the organizations were able to articulate a strategy and define tasks for achieving their goals, they had difficulty understanding the difference between *success indicators* (indicators used to determine if the goals have been met) and *progress indicators* (indicators used for tracking the execution of tasks). These organizations were using the indicators used for tracking the execution of tasks as a proxy for measuring if the goal had been achieved. When all the tasks had been executed, the organizations declared success—their goals were met. They did not analyze the outcome of the tasks as part of the decision process for determining if the goals have been met successfully. Execution of the defined tasks is a necessary but not sufficient condition for meeting the goal.

We used the following figure to clearly illustrate the differences in the type of indicators:

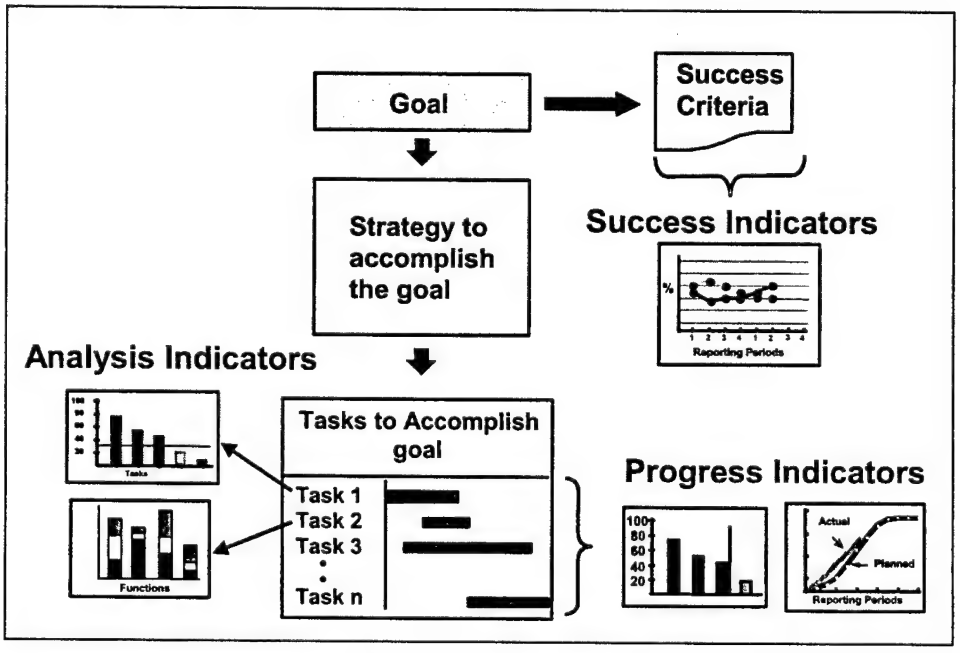


Figure 4: Types of Indicators

This figure illustrates three types of indicators:

1. **Success Indicators:** These indicators are constructed from the defined success criteria and are used to determine if the goals have been met.
2. **Progress Indicators:** These indicators are used to track the progress or execution of the defined tasks. A Gantt chart is a good example of this type of indicator. The successful execution of all the defined tasks does not necessarily guarantee that the goal has been successfully met.
3. **Analysis Indicators:** These indicators are used to assist in analyzing the output of each task. The analyses help test our assumptions about the data we are using to judge progress and success.

As seen in Figure 4, each of these indicator types has a specific use. To assist in postulating success indicators, we asked the workshop participants to think about the following questions:

- How do you know if you achieved the goal?
- How do you define success?
- How do you know if the goal has been met?

From the answers to these or similar questions, the criteria that can be used to decide if the goal has been met are identified. From the success criteria, success indicators can be postulated.

Advice: Have a clear understanding of the type and purpose of each indicator. Articulate clearly the criteria you will use to decide if the goal has been met. Do not use Progress Indicators as a proxy to Success Indicators. Use Analysis Indicators to study the data you use, in order to support accurate progress and success tracking.

5.3 Number of Indicators

Selecting the number of indicators was one of the most difficult decisions we had to make. As senior management was the audience for the indicators, the workshop participants decided the indicators should constitute a comprehensive profile that could fit on one page. The intent of this profile was to provide an overview to focus the reviewer's attention on the key issues. Other charts could be attached if necessary.

Advice: Start small and build on success. As a starting point, limit the number of indicators so that they fit on one page.

5.4 Using the Full Set of Indicators

Each indicator, taken alone, has an obvious interpretation. Shorter cycle time is good; an increase in the number of errors that a customer finds is bad, and so on. The obvious interpretation is not necessarily the correct one. Cycle time can be shortened in several ways,

some of which are clearly not desirable (e.g., skipping testing). A profile should be comprehensive, easy to understand, and force an awareness of possible hidden tradeoffs. For example, if testing is sacrificed to reduce cycle time, this may show up in the number of defects reported by the customer.

Advice: Develop a comprehensive set of indicators to detect trends and hidden tradeoffs.

6 Definitions

Definitions are critical for achieving proper interpretations of the data. Crafting a set of definitions that can be understood—not misunderstood—is one of the keys to any measurement effort. Without clear definitions, it is impossible to interpret any result in a meaningful way. We have found that the use of specialized templates and checklists enables us to collect and communicate data using a set of uniform definitions.

6.1 Indicator Definition Template

In a measurement program that encompasses multiple sites and business units, the issue of good definitions becomes more difficult and much more important. Different sites typically have different processes, business and technical environments, cultures, and assumptions. Due to the global scope of Case 1, the site personnel spoke different languages. To ensure that each unit would construct each indicator the same way using the same measures, assumptions, algorithm, etc., we developed a template for defining and documenting each indicator.

The template includes fields for

- precise objective of the indicator
- inputs
- algorithms
- assumptions
- data collection information
- data reporting information
- analysis and interpretation of results

In all our example cases, the completed templates for each indicator were collected in a measurement handbook that was distributed to each unit. A sample indicator template is provided in Figure 5. Appendix B contains a more detailed description of each field in the indicator template.

Based on feedback that we have received from organizations, the template was one of the key ingredients to success when implementing a measurement program. Organizations tend to tailor the template to fit their environment. Adding, modifying, or deleting fields, in advance of specifying a set of indicators, can help ensure that the template will be accepted and implemented by the organization.

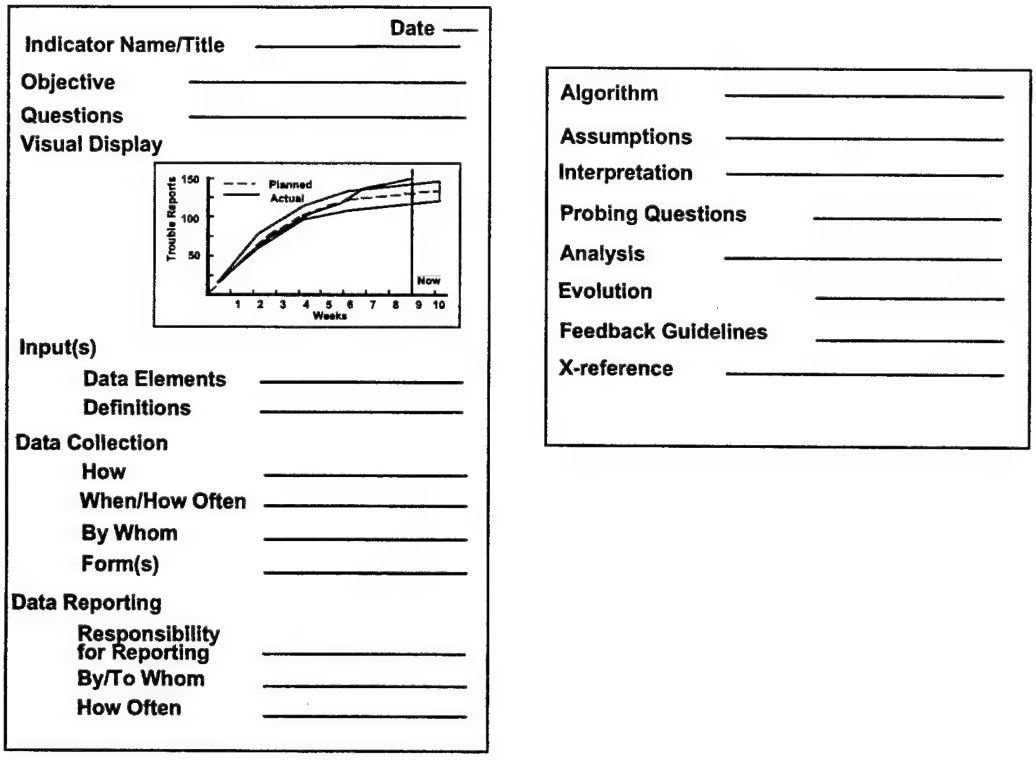


Figure 5: Indicator Template

The concept of an indicator template is not unique. Other organizations have recognized the importance of precise communication and collecting measurements based upon why they need the information rather than collecting measures because they have the capability to measure. Capt. Thomas Augustine in "An Effective Metrics Process Model" [Augustine 99] describes a form that is used to collect the information for each indicator in their metrics plan for his organization. The individual fields are very similar to those of the indicator template. The contents of the metrics collection form are shown in Table 2. Both the indicator template described in Figure 5 and the metric collection form shown in Table 2 provide information so that everyone from the collectors to the decision-makers can understand their purpose in collecting, reporting, and making decisions based on this metric [Augustine 99].

Table 2: Metrics Collection Form [Augustine 99]

Metric Title	Brief Description	
Link to Goals/Objectives	Decision(s) based on analysis	Who makes decision(s)
Who collects data	How is data collected	How often is data collected
Who reports data	How and to whom is data reported	How often is data reported
Who analyzes data	How is data to be analyzed (formulas and factors)	
Lowest acceptable value	Highest acceptable numeric values	Expected values
At what point will you stop collecting this metric		

Advice: Customize the indicator template for relevance in your environment by adding, modifying, and deleting fields as required. Define all indicators using the indicator template and use it for precise communication.

6.2 Definition Checklist

Communicating measurement definitions in clear and unambiguous terms is a non-trivial undertaking. To assist in this task, the SEI developed a series of measurement framework checklists for common software measures such as size, effort milestones, and defects. These framework documents can be downloaded via the SEI Web site at http://www.sei.cmu.edu/sema/publications.html.

The general format of the definition checklists is show in Figure 6. Each checklist contains an identification section followed by the principal attributes that characterize the object we want to measure. The values that the attribute can assume are listed for each attribute. These values must be both exhaustive (complete) and mutually exclusive (non-overlapping). The checklist also contains columns that specify if the specific value of the attribute is included or excluded in the data collected. By using a checklist of this format, the principal attributes and their values can be explicitly identified.

Identification Section			
Attribute #1 Value 1 Value 2 ⋮ Value N	Includes	Excludes	Optional
Attribute #2			
Attribute #M			

Figure 6: Definition Checklist

In all three examples, we tailored the checklists to the specific environments. In Case 1 and Case 2, we developed a set of customized checklists, based upon the SEI definition checklists to define the critical data elements. Being able to specify explicitly what attribute values were to be included and excluded in the final value for staff-hours, for example, made it possible to compare data collected from the different organizations. Figure 7 illustrates a portion of the tailored Staff-Hour Definition Checklist developed for Case 1.

Staff-Hour Definition Checklist		
Hours related to specific project	**Totals include**	**Report totals**
Activity		
Development		✓
Primary development activity	✓	
Development support activities		
Concept demo/prototypes	✓	
Tools development, acquisition, installation, & support	✓	
Nondelivered software & test drivers	✓	
Maintenance		
Non-discretionary		
Defect correction (bug fixes)	✓	✓
Other		✓
Regulatory/compliance	✓	
Release upgrade	✓	
Interface (external and internal)	✓	
Enhancements		✓
Legacy Systems	✓	
Non-Legacy Systems		
Employment Class		
Company employee		
Full time	✓	
Part time	✓	
Temporary employee	✓	
Subcontractors	✓	
Consultants	✓	

Figure 7: Adapted Staff-Hour Checklist

Advice: Use definition checklists to explicitly define your measures.

6.3 Precise Definitions

When working with multiple sites, we found (in Case 1) that there were significant differences in the assumptions being made about the start and end date of a project, as well as other key dates (milestones) in the development process. It was very difficult to combine data from individual projects into a comprehensive view for the entire enterprise or larger unit unless there was a consistent definition of key dates in the development. We developed a checklist that specified exactly what constitutes the start and end dates of a project as well as other milestones. To develop this checklist, each site presented how they precisely defined a project's start and end date. The data from all the sites were consolidated, and common definitions for these dates were developed. Figure 8 illustrates how the checklist that specifies project start and end dates was developed.

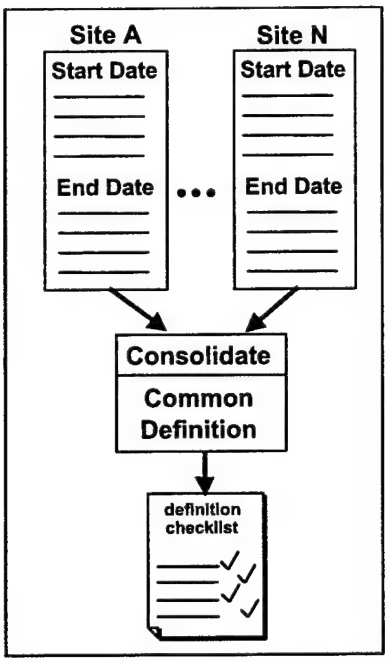


Figure 8: Developing a Checklist for a Project's Start and End Dates

In Case 1, English was not the native language of most of the participants. We found it extremely useful to use graphics to illustrate concepts and definitions. Graphics were also extremely useful to prevent misunderstandings when working with the different business units in Case 2. Figure 9 is an example of one of the graphics used with Case 1.

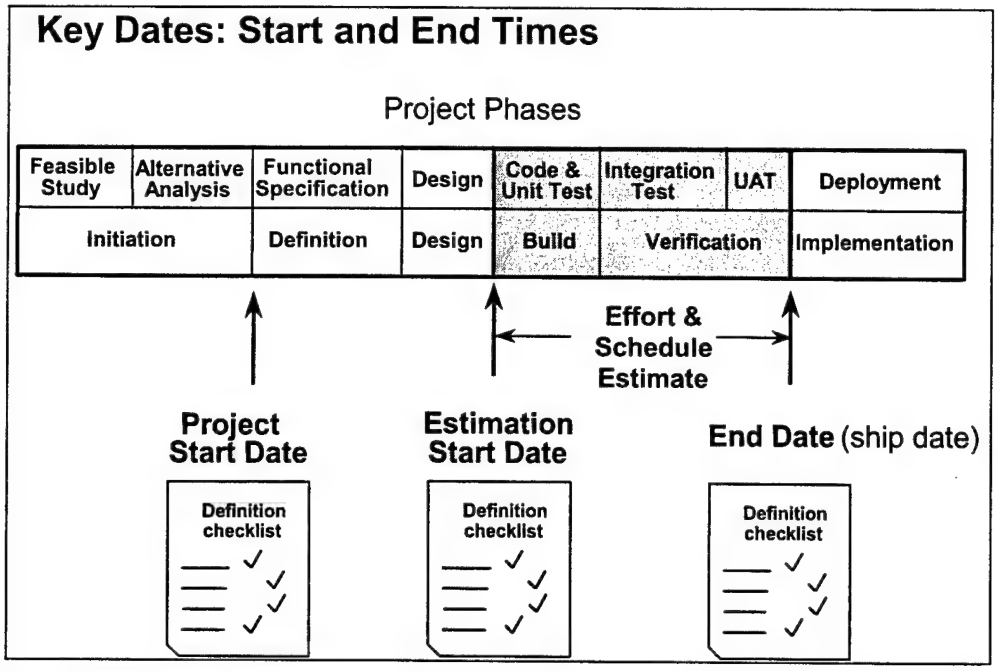


Figure 9: Date Definition Checklists

Advice: Use specialized templates, checklists, and graphics to disseminate unambiguous information that precisely defines the inputs for the measurement program.

7 Data Elements or Measures

7.1 Use of the Data Elements: Etiquette

Using data appropriately is one of the keys to getting a metrics program off the ground, motivating people, and sustaining commitment. If individuals see the metrics as tools to help them succeed, they are likely to rally to the cause. If, on the other hand, they feel robbed of respect, treated like a tool rather than a person, stiff resistance will be the result. To ensure that measures are used appropriately, honor the following three principles:

1. Never allow anyone in the organization to use metrics to measure individuals.
2. Specify how the data are being used by relating them to strategy and providing regular feedback to staff.
3. Have clear rules about who has access to specifics of data, and clear hand-offs when it passes from private status to public. Some data should only be made available to specific individuals; while other data can be accessed at the project- or organizational-level. The table below shows the breakdown suggested by Robert Grady [Grady 92].

Table 3: Data Access Recommendations

Individual	Project Team	Organization
Defect rates (by individual)	Defect rates (team)	Defect rates (by project)
Defect rates (by module)	Module size	Size (by product)
Defect rates (under development)	Estimated module size	Effort (by project)
Number of compiles	Number of re-inspections	Calendar times
	Defects per module (prerelease)	Defects per module (post release)
		Effort per defect (average)

Advice: Pay close attention to privacy issues pertaining to who can see what portion of the data.

7.2 Collection of Data

Culture may aid or hinder the implementation of a measurement program. In Case 1, (Measurement Across a Global Enterprise), cultural differences among the sites caused us to expend considerable resources. At one site, measurement was engrained in all activities. At others, our request for effort data was taken as an insult. Considerable time and effort was expended explaining the benefit of collecting and sharing this information. Each time a new individual came on board, we had to re-address this cultural issue.

Advice: Culture is a major issue. Plan to address it early and throughout the implementation. Respect the needs of people involved, and work collaboratively.

7.3 The 100% Solution May Not Be Feasible

Trying for the perfect solution that will satisfy all participants may be a futile endeavor. It may be impossible to obtain agreement on all issues by all the participants. A good example of this problem was our solution for the unit of size for Case 1. A number of candidate solutions were proposed that included Function Points, source lines of code (SLOC), as well as several "proxy" measures such as screens and functions. All have their advantages and disadvantages. Since the primary applications being developed by this organization are database and report-intensive information systems in a variety of languages, Function Points seemed a natural choice. On the other hand, SLOC has the advantage of being relatively inexpensive to collect reliably, given a careful definition of what counts as a line of code and which lines to count. A considerable amount of time and effort were expended defending each particular candidate.

To come to some kind of resolution, we conducted a survey to determine how much software existed in each language and how much would be constructed in each language in the next year. The survey showed that we could count size in SLOC with existing code counters for around 80% or more of the software being developed. Also, using available code counters is a relatively inexpensive way to collect size information when compared to Function Point analysis. This information allowed us to come to the initial solution of using SLOC to determine the size of 80% of the software. The remaining 20% would be addressed later. We also decided to support experimentation with Function Points and other "proxy" measures to get a better sense of the expense and potential advantages.

Advice: When there is no consensus on how to do something, (e.g., measure size) take as your initial position the least costly of the adequate solutions available. Then experiment on a limited basis with other solutions to see if they demonstrate added value. Since no solution will please everyone under these circumstances, adopting any single solution will require some of the adopters to implement something in which they don't fully believe. The lower the cost, the more likely they are to go along. It may be impossible to obtain the 100% solution, 80% may be good enough.

8 Lessons Learned

8.1 Pilot Implementations

Pilot implementations allow us to test the feasibility and robustness of definitions, checklists, templates, and procedures developed to implement the measurement program as well as to develop the operational aspects of:

- forms for collecting and recording data
- data storage and access tools
- who will collect, store, and access data
- tools to aid in collection and analysis
- roll up procedures
- training

The pilot implementations enabled us to identify a number of problem areas. One identified problem was related to what constituted the end of the project. From the checklist shown in Figure 10 for Case 1, we can see that the end of the project was signaled by customer sign-off at the end of User Acceptance Testing (UAT). During the pilot implementation, we found that some customers would deploy the software and never execute a sign-off at the end of UAT.

As a result of the pilot, we had to modify the checklist so that a measurable and observable event that all could agree on would signal the end of the project.

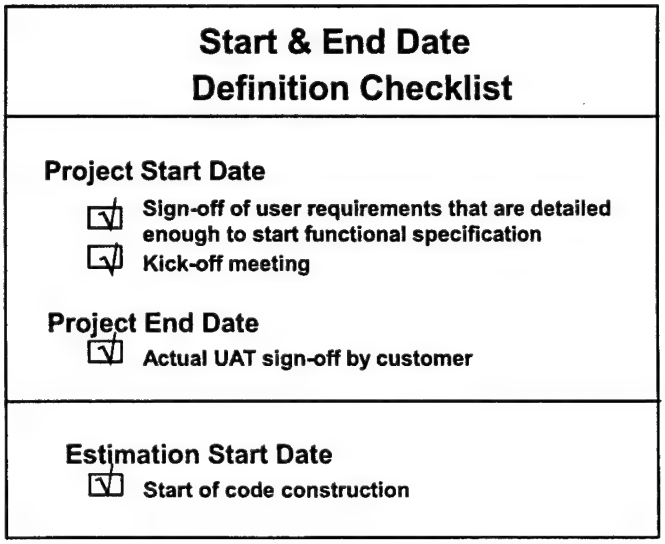


Figure 10: Example of a Start and End Date Checklist

Advice: Use pilot implementations to verify feasibility and to test definitions, checklists, and templates.

8.2 Implementation Time Frame

Implementing a measurement program across many different business units was a tedious, time-consuming process. In a number of cases, major reorganizations and retirement of key management stakeholders during the pilot implementations hampered progress. The new management had other priorities, which made it difficult to maintain project momentum. Since the basic indicators provided considerable value to the individual business units, the collection and refinement of the indicators, templates, and forms continued.

Advice: Recognize that implementation of a measurement program may take a long time and that management can have a short-term window. Therefore, plan to show some short-term successes before management change. Start small and build upon success.

8.3 Automation/Tools

In general, you should automate the collection of the raw data as much as possible. This will reduce the effort required to collect the raw material on which the measurement program is built. Automation focused on simplifying the recording of primitive data elements will tend to be more beneficial at first. Elaborate data analysis and presentation tools are best selected after the stakeholders have an opportunity to explore the amount of decision support available in the data collected. The ability to refine or add primitive data elements will be more beneficial at first than the ability to perform a complex analysis or draw an intricate display. Effective communication tools must be selected after the nature of the communication is well understood.

Advice: Make the tool fit the process, not the other way around. Maximize yield of relevant information, while minimizing data collection effort.

8.4 Using Analysis Indicators

In Case 3, the rate of implementation for system change requests had decreased dramatically in recent times, and quality concerns expressed by the customer had increased.

Questions about the relationship of productivity in the implementation of change requests and quality led to analyzing the processes used to accomplish changes to the system. Analysis determined that each system release contained changes accomplished using three different processes:

1. using the standard process
2. using an abbreviated process
3. using the process for emergency fixes

Analysis of the delivery rate for change packages revealed the following pattern of package types shown in Figure 11. Figure 11 shows the number of change packages implemented by each of the three changes processes used for system releases. As can be seen, the processes used to implement changes packages have changed dramatically in recent times. A large number of emergency fixes are seen in most releases starting with release 3. The number of changes being released with the abbreviated process consistently exceeds the rate of changes released using the standard process.

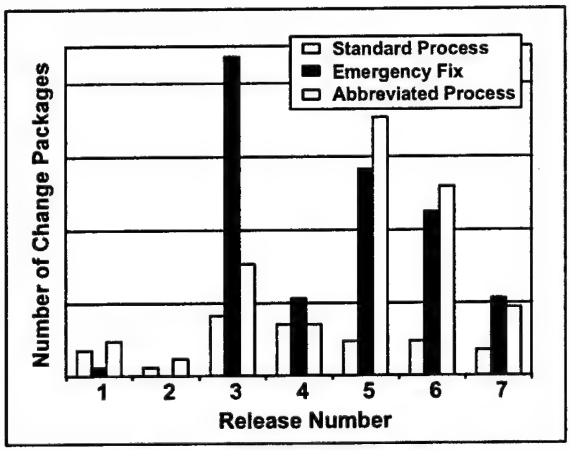


Figure 11: Changes per Release Number

The differentiating factor among these processes is the degree of formality and amount of time allocated to the process of analyzing the change and obtaining agreement to the proposed solution. The standard process requires approval of a very time consuming Change Control Board, while the approval process of the other two processes was much less formal hence less time consuming.

The relationship between quality and the change process used must be understood before modifying the standard process. Whether the occurrence of quality issues leads to unpredictable schedules or the compression of schedules leads to quality problems must be understood. Changes made to the standard process and the abbreviated process may independently affect the schedule or quality performance or both. Improving the standard process (assuming that it will have a uniform effect on performance) requires further analysis. Also further analysis must be performed to address the quality concerns expressed by the customer.

Advice: Look at your data, and test your assumptions. Don't be afraid to revise your intuition based on evidence.

8.5 Motivating the Wrong Behavior

Measurement stakeholders, determined to find useful information to guide decisions, can be lead astray by context-dependent information presented out of context. In the example from

Case 3 described above, one initial reaction to the situation was a suggestion to eliminate the abbreviated process—in an effort to enforce the standard process. However, this would potentially limit the productivity of the project even further, unless some effort to improve the usability of the standard process is undertaken. Merely eliminating (an apparently productive) avenue for releasing products is an apparent "near term fix" with potentially counterproductive long-term consequences.

Data provided to wider audiences can be censored or distorted when the intentions of data recipients are not understood (or trusted). Defect data from inspections are frequently subject to bias due to effort expended by some to perform pre-review quality checks. However, others spend less time (or no time) preparing for the measurement point. While the performance measures indicate improved performance, the prevalence of undocumented ways of doing work exceeds the organization's ability to balance the workload. The problem compounds itself as the data used become more and more distorted, and the decisions justified by the data become less and less credible.

Matters of trust and motivation aren't the only sources of distortion in data that motivate the wrong behavior. Conflicting goals for performance can frequently lead to tradeoffs in accuracy, which constrain the performance of projects with unintended consequences. Organizations struggling to manage "unit cost" often find themselves defining and re-defining a narrowly focused, context-independent, performance index driven by arbitrary definitions and counting rules negotiated in committees. The inclusion or exclusion of various effort categories such as overtime, vacation time, project management, quality assurance, and rework can perpetuate an unrealistic expectation for performance. When one group of roles is motivated by minimizing an arbitrary measure, their influence on another group of roles in the organization can lead to a type of shell game that maximizes the appearance of performance in one respect, while actively sabotaging the reliability of information available.

In many cases, the choice not to collect and examine some data may actually cause people to use the data for more appropriate purposes. Respecting the ownership of data by the people who can most directly act on it will reduce the occurrence of unintended consequences due to conflicting perspectives on what the data mean.

Advice: Beware of unintended consequences, and the perspectives of different stakeholders. Make the right thing to do the easy thing to do.

8.6 Summary of Lessons Learned

We have used the Goal-Driven Software Measurement methodology to implement measurement programs in a large number of organizations. The three example cases had different goals for their measurement programs. The solutions to the problems and

impediments encountered, the artifacts developed (such as templates and checklists), and the lessons learned will provide insight to others trying to implement a measurement program.

The following is a summary of advice to those currently implementing or considering implementing a measurement program:

Summary of Advice
Use the GQ(I)M methodology to identify your indicators and measures to ensure traceability back to the business goals.
Clearly specify the goal that is being addressed.
Maintain traceability from the indicators back to the business goals. If questions arise later about intent, you will be able to look back to the origins and provide implementation decisions that are consistent with your business objectives.
Have a clear understanding of the type and purpose of each indicator. Articulate clearly the criteria you will use to decide if the goal has been met. Do not use Progress Indicators as a proxy to Success Indicators. Use Analysis Indicators to study the data you use, in order to support accurate progress and success tracking.
Start small and build on success. As a starting point, limit the number of indicators so that they fit on one page.
Develop a comprehensive set of indicators to detect trends and hidden tradeoffs.
Customize the indicator template for relevance in your environment by adding, modifying, and deleting fields as required. Define all indicators using the indicator template and use it for precise communication.
Use definition checklists to explicitly define your measures.
Use specialized templates, checklists, and graphics to disseminate unambiguous information that precisely defines the inputs for the measurement program.
Pay close attention to privacy issues pertaining to who can see what portion of the data.
Culture is a major issue, plan to address it early and throughout the implementation. Respect the needs of people involved, and work collaboratively.
When there is no consensus on how to do something, (e.g., measure size) take as your initial position the least costly of the adequate solutions available. Then experiment on a limited basis with other solutions to see if they demonstrate added value. Since no solution will please everyone under these circumstances, adopting any single solution will require some of the adopters to implement something in which they don't fully believe. The lower the cost, the more likely they are to go along. It may be impossible to obtain the 100% solution, 80% may be good enough.
Use pilot implementations to verify feasibility and to test definitions, checklists, and templates.
Recognize that implementation of a measurement program may take a long time and that management can have a short-term window. Therefore, plan to show some short-term successes before management moves on. Start small and build upon success.

Make the tool fit the process, not the other way around. Maximize yield of relevant information, while minimizing data collection effort.
Look at your data and test your assumptions. Don't be afraid to revise your intuition based on evidence.
Beware of unintended consequences and the perspectives of different stakeholders. Make the right thing to do the easy thing to do.

8.7 Conclusion

When implementing a measurement program, pay special attention to the lessons we learned and to the artifacts we developed such as templates and checklists. They may assist you in becoming a success data point in Howard Rubin's database of companies that have implemented a measurement program.

References

[Augustine 99] Augustine, Thomas & Schroeder, Charles. "An Effective Metrics Process Model." *CrossTalk 12*, 6, (June 1999) 4-7.

[Basili 84] Basili, V. & Weiss, D. "A Methodology for Collecting Valid Software Engineering Data." *IEEE Transactions on Software Engineering 10*, 6 (November, 1984): 728-738.

[Basili 88] Basili, Victor R. & Rombach, H. Dieter. "The TAME Project: Towards Improvement-Oriented Software Environments." *IEEE Transactions on Software Engineering 14*, 6 (June 1988): 758-773.

[Basili 89] Basili, Victor R. "Using Measurement for Quality Control and Process Improvement." *Proceedings of the Software Engineering Process Group Conference.* Pittsburgh, Pennsylvania, June 21-22, 1989.

[Briand 96] Briand, L.; Differding, C.M.; & Rombach, H.D. "Practical Guidelines for Measurement-Based Process Improvement." *Software Process—Improvement and Practices 2*, 4 (December, 1996): 253-280.

[Florac 92] Florac, William A. et al. *Software Quality Measurement: A Framework for Counting Problems and Defects.* (CMU/SEI-92-TR-022, ADA 258 556). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, September 1992. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.022.html

[Goethert 92] Goethert, Wolfhart B. et al. *Software Effort & Schedule Measurement: A Framework for Counting Staff-Hours and Reporting Schedule Information.* (CMU/SEI-92-TR-21, ADA 258 279). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, September 1992. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.021.html

[Goethert 97] Goethert, Wolfhart B.; Herbsleb, James; & Pasternack, Gerald. "Software Measurement Across a Global Enterprise: An Experience Report." *Proceedings of the Software Engineering Process Group Conference.* San Jose, California. March 17-20, 1997.

[Grady 92] Grady, Robert, *Practical Software Metrics for Project Management and Process Improvement*, Englewood Cliffs, N.J.: Prentice Hall, 1992.

[Knox 93] Knox, S. T. "Modeling the Cost of Software Quality." *Digital Technical Journal 5*, 4 (1993): 9-17.

[Park 96] Park, Robert; Goethert, Wolfhart and Florac, William. *Goal-Driven Software Measurement – A Guidebook.* (CMU/SEI-96-HB-002). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, August 1996. http://www.sei.cmu.edu/publications/documents/96.reports/96.hb.002.html

[Park 92] Park, Robert E. et al. *Software Size Measurement: A Framework for Counting Source Statements.* (CMU/SEI-92-TR-020, ADA 258 304). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, September 1992. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html

[Pitts 97] Pitts, David R. "Metrics: Problem Solved?" *CrossTalk 10*, 12 (December 1997): 28-30.

[Rifkin 91] Rifkin, Stan & Cox, Charles. *Measurement in Practice.* (CMU/SEI-91-TR-016) Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, July 1991. http://www.sei.cmu.edu/publications/documents/91.reports/91.tr.016.html

[Rombach 89] Rombach, H. Dieter & Ulery, Bradford T. "Improving Software Maintenance Through Measurement." *Proceedings of the IEEE 77,* 4 (April 1989): 581-595.

[Rubin 92] Howard Rubin Associates, Inc. "The Making Measurement Happen Workshop." *Proceedings of the 3rd International Conference on Applications of Software Measurement.* La Jolla, California. November 15-19, 1992.

[Zubrow 98] Zubrow, David. "Measurement With a Focus: Goal-Driven Software Measurement?" *CrossTalk 11*, 9 (September 1998): 24-26.

Appendix A: Indicators For Case 1

Indicators for an Enterprise-Wide Measurement Program

Figure 12 shows a version of the Enterprise Profile created for Case 1.[1] They are grouped by legend, so that, for example, all four charts that separate the data into small, medium, and large projects are grouped together with the legend for project size category. Following is a brief description of each indicator. These descriptions are based on the needs of the organization discussed in this paper, and may substantially differ from what other organizations would need. These are provided as examples, not as advice on what the reader should measure.

Schedule Predictability. This indicator is designed to answer questions about an enterprise's ability to plan and deliver the product on schedule. It shows schedule deviation expressed as a percentage of planned project duration. Both early and late delivery are treated as deviations and counted in the same way.

Effort Predictability. In order to improve cost estimation and the ability to bring projects in on budget, this indicator shows cost deviation (underruns and overruns are treated identically) expressed as a percentage of the original cost estimate.

Cycle Time. This indicator shows calendar days per size unit, which is used to track improvements in getting products to market as quickly as possible.

Quality. This indicator has two components. One is defects per size unit at User Acceptance Testing (UAT), which gives some indication of the quality of the development and testing process and which should also be a leading indicator of the quality of the software in the field. The number of high-priority field defects is tracked both because it is an important component of quality and because it aids in interpreting UAT defect densities. UAT defects could go down either because the quality is good going into UAT or because UAT is ineffective. If the latter is true, field defect densities should go up in subsequent periods.

Maintenance Effort. Like many organizations struggling with huge volumes of legacy code, this organization wants to reduce expenditures on maintenance to make resources available for new development. This indicator tracks the percentage of effort-hours on enhancements and on non-discretionary spending (i.e., bug fixes, regulatory changes, interface changes, etc.). It also tracks high-priority open requests to guard against the possibility that reduced

[1] The data values are for illustrative purposes only, and do not represent real data.

maintenance spending is being accomplished by failing to service high-priority requests that should not be neglected.

Customer Satisfaction. This indicator tracks two components of customer satisfaction: satisfaction with the implemented solution and satisfaction with the working relationship with the implementing team.

Cost of Quality (COQ). This analysis breaks down overall costs (in effort-hours) into four categories. We modified the approach used by Crosby to fit the needs of this organization. The categories of cost that we used are:

- rework: total hours for fixing defects discovered prior to release, including the cost of re-inspecting and retesting
- appraisal: total hours for inspecting and testing (except when those hours are part of rework)
- prevention: total hours for defect prevention activities, such as Pareto analysis
- performance: costs that are not one of the above (e.g. effort associated with building the product).

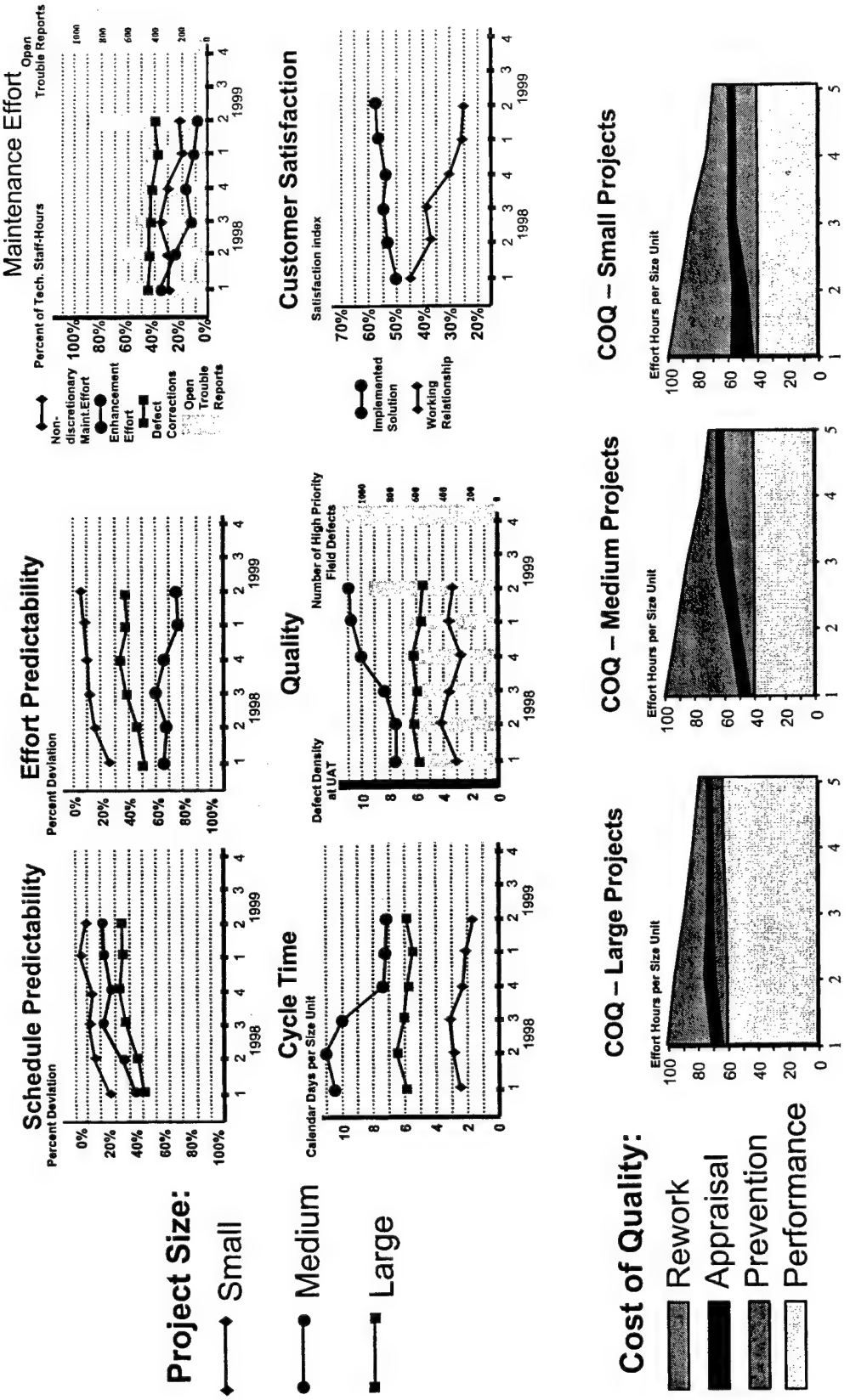


Figure 12: Enterprise Profile Example (Data values are for illustrative purposes only)

Appendix B: Indicator Definition Template

Indicator Template

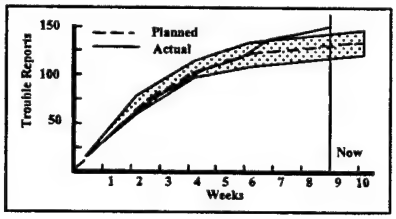
A template that may be used to describe the indicators is shown below.

INDICATOR TEMPLATE

Objective _____

Questions _____

Visual Display



Input(s)

Data Elements _____

Responsibility for Reporting _____

Form(s) _____

Algorithm _____

Assumptions _____

Interpretation _____

X-reference _____

Probing Questions _____

Evolution _____

Figure 13: Indicator Template

INDICATOR TEMPLATE

Date _____

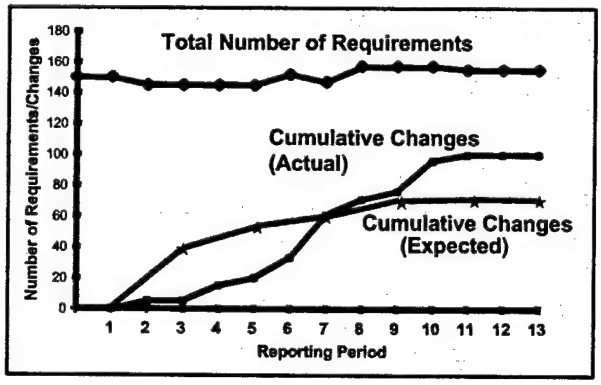
Indicator Name/Title: _____

OBJECTIVE: The description of the objective or purpose of the indicator.

QUESTIONS: Question or questions the user of the indicator is trying to answer. Examples: Is the project on schedule? Is the product ready to ship? Should we invest in moving more software organizations to CMM maturity level 3?

VISUAL DISPLAY

A graphical view of the indicator.



INPUT(S)

Data Elements

A list of the data elements that are used in the production of the indicator. The description needs to be sufficiently precise so that different individuals in different environments can reliably generate comparable numbers in the same units.

Responsibility for Reporting

Who has responsibility for reporting the data.

Forms

If there is a standard form for data collection, it should be referenced here and information should be provided about where to obtain the latest version.

ALGORITHM

Specify the algorithm or formula required to combine data elements to create input values for the display. It may be very simple, such as Input1/Input2, or it may be much more complex. It should also include how the data are plotted on the graph.

ASSUMPTION

Identify any assumptions about the organization, its processes, life cycle models, and so on that are important conditions for collecting and using this indicator.

INTERPRETATION

What different values of the indicator mean. This section should make it clear how the indicator answers the questions in the "Questions" field above. It should also provide any important cautions about how the data could be misinterpreted and measures to take to avoid misinterpretation.

X-References

If the values of other defined indicators will influence the appropriate interpretation of the current indicator, refer to them here.

Probing Questions

List questions that delve into the possible reasons for the value of an indicator, whether performance is meeting expectations or whether appropriate action is being taken.

Evolution

Describe specific ideas about how the indicator can be improved over time, especially as more historical data accumulates (e.g., by comparison of projects using new processes, tools, environments with a baseline; using baseline data to establish control limits around some anticipated value based on project characteristics).

Modified Indicator Template

A modified template that may be used to describe the indicators is shown below. This indicator template illustrates how the indicator template may be tailored to add other specific data requirements. The new fields are shown in dark gray.

MODIFIED INDICATOR TEMPLATE

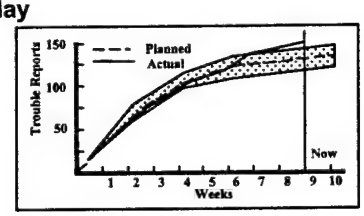
Date

Indicator Name/Title

Objective

Questions

Visual Display



Input(s)

- Data Elements
- Definitions

Data Collection

- How
- When/How Often
- By Whom
- Form(s)

Data Reporting

- Responsibility for Reporting
- By/To Whom
- How Often

Algorithm

Assumptions

Interpretation

Probing Questions

Analysis

Evolution

Feedback Guidelines

X-reference

Figure 14: Modified Indicator Template

Date _____

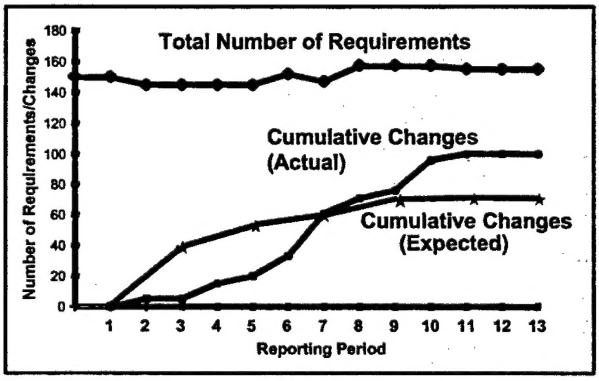
Indicator Name/Title: _____

OBJECTIVE: The description of the objective or purpose of the indicator.

QUESTIONS: Question or questions the user of the indicator is trying to answer. Examples: Is the project on schedule? Is the product ready to ship? Should we invest in moving more software organizations to CMM maturity level 3?

VISUAL DISPLAY

A graphical view of the indicator.



INPUTS

Data Elements	**Definitions**
A list of all the data elements used in the production of the indicator.	The precise definitions of the data elements or pointers to where the definition can be found.

DATA COLLECTION

How

A description of how the data are to be collected.

When/How Often

When are the data to be collected and how often.

By Whom

Specify who is going to collect the data. This could be an individual, an office, etc.

Forms

If there is a standard form for data collection, it should be referenced here and information should be provided about where to obtain the latest version.

DATA REPORTING

Responsibility for Reporting

Specify who has responsibility for reporting the data.

By/To Whom

Gives some indication of who is going to do the reporting and to whom the report is going. This may be an individual or an organizational entity.

How Often

Specify how often the data will be reported (daily, weekly, monthly, as required, etc.).

ALGORITHM

Specify the algorithm or formula required to combine data elements to create input values for the display. It may be very simple, such as Input1/Input2, or it may be much more complex. It should also include how the data are plotted on the graph.

ASSUMPTION

Identify any assumptions about the organization, its processes, and life-cycle models, etc. that are important conditions for collecting and using this indicator.

ANALYSIS

Specify what type of analysis can be done with the information.

INTERPRETATION

Explain what different values of the indicator mean. This section should make it clear how the indicator answers the questions in the "Questions" field above. It should also provide any important cautions about how the data could be misinterpreted and measures to take to avoid misinterpretation.

PROBING QUESTIONS

Questions that delve into the possible reasons for the value of an indicator, whether performance is meeting expectations, or whether appropriate action is being taken.

EVOLUTION

Specific ideas about how the indicator can be improved over time, especially as more historical data accumulates (e.g., by comparison of projects using new processes, tools, environments with a baseline; using baseline data to establish control limits around some anticipated value based on project characteristics).

FEEDBACK GUIDELINES

X-REFERENCES

If the values of other defined indicators will influence the appropriate interpretation of the current indicator, refer to them here.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	November 2001	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Experiences in Implementing Measurement Programs	F19628-00-C-0003

6. AUTHOR(S)

Wolfhart Goethert, Will Hayes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2001-TN-026

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Despite significant improvements in implementing measurement programs for software development in industry, data collected by Rubin Systems show that a large percentage of metrics programs fail. This technical note describes some useful lessons learned at a number of organizations that have implemented measurement programs using the Goal-Driven Software Measurement methodology. It includes a description of the methodology, a discussion of the challenges, obstacles, and their solutions, an initial set of indicators and measures, as well as some artifacts (such as templates and checklists) that we have found to enable successful implementations. The main motivation of this technical note is to provide some practical advice and guidelines for planning and implementing measurement programs.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Measurement implementation, goal-question-metric (GQM), Indicator template, Measurement program, Measurement, Measurement experiences	46

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL